VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

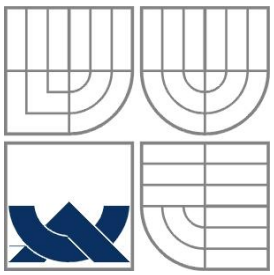
WEBOVÁ SLUŽBA JAKO ROZHRANÍ PRO GOOGLE CALENDAR

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

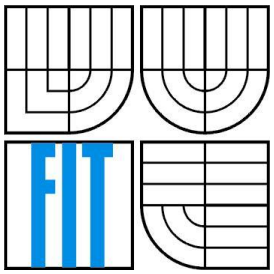
AUTOR PRÁCE
AUTHOR

MARTIN PAULÍČEK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

WEBOVÁ SLUŽBA JAKO ROZHRANÍ PRO GOOGLE CALENDAR

WEB SERVICE AS AN INTERFACE FOR GOOGLE CALENDAR

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE — MARTIN PAULÍČEK
AUTHOR

VEDOUCÍ PRÁCE — ING. PETR WEISS
SUPERVISOR

BRNO 2008

Abstrakt

Úkolem této práce bylo vytvoření webové služby, která by zprostředkovala propojení s webovým kalendářem Google Calendar. Tato služba by měla být dále propojena s registrem služeb UDDI. Další částí práce bylo vytvoření klientské aplikace komunikující s touto webovou službou a registrem služeb. Klientská část by měla zároveň sloužit pro demonstrační účely ukazující použití webové služby a jejich komunikaci.

Klíčová slova

Webová služba, WS, Webové rozhraní, Komunikační protokol webové služby, SOAP, Popis webové služby, WSDL, Registr webových služeb, UDDI, Webový kalendář, Google Calendar, HTTP, XML.

Abstract

The objective of this project was to create a web service that provides a connection to the web calendar named Google Calendar. This service should co-operate with an UDDI service registry. Another part of this project was creation of a client application that should communicate with the web service and the service registry. The client application should also serve as a demonstration application for presenting principles of web services and their communication.

Keywords

Web service, WS, Web interface, Web service communication protocol, SOAP, Web service description, WSDL, Web service registry, UDDI, Web calendar, Google Calendar, HTTP, XML.

Citace

Martin Paulíček: Webová služba jako rozhraní pro Google Calendar, bakalářská práce, Brno, FIT VUT v Brně, 2008

Webová služba jako rozhraní pro Google Calendar

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Petra Weisse.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Martin Paulíček
28.4.2008

Poděkování

Rád bych zde poděkoval Ing. Petru Weissovi za konzultace a odborné vedení při tvorbě této práce.

Obsah

Úvod

Základem každodenní práce na internetu je komunikace. Právě zde se nachází uplatnění webových služeb. Webová služba (dále jen WS) slouží jako univerzální standardizované rozhraní, pomocí kterého můžeme komunikovat s protější stranou přes internet. Byla navrhnuta a je používána při strojové výměně informací po síti. Právě toto rozhraní umožní vytvořit přístup k webové aplikaci Google Calendar (dále jen GC), které bude nezávislé platformou i jazykem na klientském programu.

Google Calendar patří k jednomu z mnoha webových softwarů společnosti Google. Již podle názvu uhodneme, že se jedná o program pracující s kalendářem. Aplikace umožňuje vytvořit a spravovat vlastní i cizí virtuální kalendáře. Kalendář nám nabízí veškerou funkčnost, kterou požadujeme od klasického kalendáře.

V této bakalářské práci se zaměříme na tvorbu výše zmíněného rozhraní k správě celé aplikace Google Calendar. Rozhraní bude navrženo podle standardizovaných protokolů webových služeb. Dále vytvoříme ukázkového klienta, který bude umět pracovat s tímto rozhraním, a tedy bude schopen dělat všechny úkony, které jsou možné z webového prostředí. Klient bude dále zaměřen na výuku WS. Bude tedy pracovat se serverovou částí a přitom nabízet podrobný náhled do probíhající komunikace WS.

Nejdříve se seznámíme s teorií webových služeb. V první kapitole se dozvíme, co je webová služba, z jakých částí se skládá a jak probíhá její komunikace. Probereme architekturu i bezpečnost webových služeb a vysvětlíme pojmy SOAP, WSDL a UDDI.

Poté v druhé kapitole se zaměříme na aplikaci Google Calendar. Povíme si, jaké jsou její možnosti. Zmíníme uživatelskou část aplikace v podobě webového rozhraní. Budeme se věnovat i vývojové stránce, přesněji aplikačnímu rozhraní, které umožňuje vývoj jiných aplikací pracujících se službou GC.

Po hlavním úvodu do problematiky WS a GC budou následovat kapitoly vztahující se ke komunikaci WS. Jedná se o kapitolu transportního protokolu a jazyku posílaných zpráv. Jako transportní protokol WS je využíván protokol HTTP, který bude popsán v třetí kapitole. V čtvrté kapitole můžeme najít popis XML jako jazyku používaného ve zprávách WS.

Následovat budou kapitoly o implementaci a návrhu. Pátá kapitola bude obsahovat návrh celé aplikace. Nejdříve probereme serverovou část, tedy rozhraní aplikace GC v podobě webové služby. Následně si popíšeme klientskou část aplikace. Implementaci navrhnutého řešení najdeme v kapitole šesté. Opět hlavní rozdělení bude na klientskou a serverovou část.

Na závěr celé práce bude uveden konečný stav projektu. Najdeme zde jeho zhodnocení, nedostatky a možná vylepšení pro další vývoj.

1 Webové služby

V této kapitole bude probrána problematika webových služeb. V první podkapitole se seznámíme s WS jako celkem. Poté budou probrány části webových služeb. Následující druhá podkapitola je zaměřena na protokol SOAP neboli zprávy, které jsou přenášeny při komunikaci. Dále v třetí podkapitole následuje popis rozhraní, který je specifikován pomocí WSDL. Nakonec v poslední čtvrté podkapitole si povíme o registru webových služeb tzv. UDDI.

1.1 Úvod do webových služeb

WS slouží jako univerzální prostředek pro komunikaci přes internet. WS byly vytvořeny pro strojovou výměnu informací, kdy při komunikaci mezi klienty není třeba žádné uživatelské interakce. WS jsou založeny buď na vzdáleném volání procedur (Remote Procedure Call), na architektuře SOA (Service-oriented architecture) nebo REST (REpresentational State Transfer). Všechny tři přístupy budou ukázány v kapitole 1.1.2 o architektuře WS.

Kvůli možné nejednotnosti rozhraní byla vydána roku 2000 specifikace WS organizací W3C (World Wide Web Consortium). Specifikace definuje dva standardy. Prvním je protokol SOAP, který určuje, jak mají zprávy při komunikaci vypadat. Druhým standardem je WSDL, který popisuje, jak má vypadat rozhraní pro komunikaci. Dále do WS patří registr webových služeb. K tomuto účelu jsou využívány servery UDDI, které umožňují zaregistrování a vyhledávání webových služeb.

1.1.1 Definice webové služby

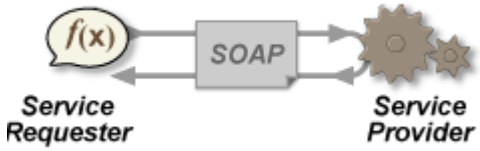
Podle organizace W3C [3] můžeme webovou službu definovat jako softwarový systém určený k spolupráci mezi stroji přes síť. Jeho rozhraní je popsáno v strojově zpracovatelném formátu (výslovně WSDL). Ostatní zařízení pracují s webovou službou, jejíž popis je předepsaný pomocí SOAP zpráv. Tyto zprávy jsou typicky přenášené protokolem HTTP s XML obsahem splňující ostatní webové standardy.

1.1.2 Architektura

Webová služba může být implementována jako vzdálené volání procedur (RPC), jako koncept SOA nebo REST. Nejdříve si popíšeme architekturu vzdáleného volání procedur a následně bude ukázána architektura SOA. Nakonec si vysvětlíme architekturu REST.

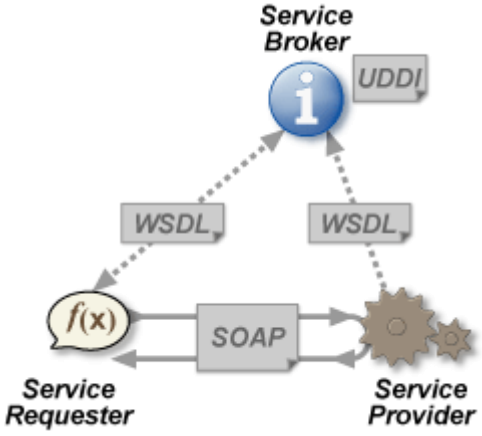
Architektura RPC bude nejlépe zřetelná z obrázku č. 1. Zde je možné vidět pouze dva aktéry, kde první z nich zastává roli klienta (Service Requester) dotazujícího se pomocí SOAP zpráv na provedení určité procedury (nebo metody). Druhý funguje jako server (Service Provider) poskytující klientovi procedury (metody) k zavolání. Výsledný děj vypadá takto: Procedura (metoda) je zavolána

klientem, zpracuje se na serveru a nakonec po jejím úspěšném provedení se pošle výsledek zpět klientovi.



Obrázek č. 1 – Vzdálené volání procedur (převzato z [9])

Architekturu SOA si můžeme prohlédnout na následujícím obrázku č. 2. Na obrázku se vyskytují nyní 3 účastníci, jedná se o klienta (Service Requester), server (Service Provider) jako u architektury RPC. Přibývá ještě UDDI registr (Service Broker), který uchovává informace o serverech s WS. Komunikace mezi klientem a serverem probíhá opět pomocí SOAP zpráv. Vzniká nám další typ komunikace mezi klientem a registrem a mezi serverem a registrem. Komunikace je zaměřená na popis webové služby WSDL. Celá komunikace probíhá takto: Server s WS zaregistruje svůj popis WSDL v registru. Teprve poté může klient pracovat s jeho WS. Klient se dotáže registru na hledanou službu, registr vrátí klientovi popis služby WSDL nebo jeho adresu. Dále probíhá komunikace jako u architektury RPC.



Obrázek č. 2 – Service-oriented architecture (převzato z [9])

Poslední členem této skupiny je REST (REpresentational State Transfer). Tato architektura je založená na myšlence webového zdroje, který je identifikovatelný pomocí své URI adresy. Zdroj je reprezentován například XML dokumentem. Možné operace se odvíjí od čtyř základních HTTP metod GET, POST, PUT a DELETE. Pomocí těchto operací můžeme určit, co se má provést za akci s určitým zdrojem (URI adresou). Příkladem může být zjištění teploty v určitém městě. U SOAP metody bychom zadali město jako parametr SOAP dotazu, zde přidáme jméno města do URI adresy. Využijeme operaci GET pro přečtení teploty a jako cestu ke zdroji uvedeme zdroj s přidaným jménem města např.: http://www.teplota.cz/Brno. Výsledkem operace bude XML dokument se jménem města a jeho teplotou.

1.1.3 Technologie

Webové služby se skládají z tří základních technologií:

- SOAP (Simple Object Access Protocol) – protokol sloužící ke komunikaci, tedy výměně XML zpráv přes počítačovou síť (standardizován organizací W3C)
- WSDL (Web Service Description Language) – jazyk pro popis rozhraní webové služby ve formátu XML (standardizován organizací W3C)
- UDDI (Universal Description, Discovery and Integration) – registr umožňující registraci a vyhledávání webových služeb

1.1.4 Komunikace

Komunikace probíhá pomocí protokolu SOAP. Jedná se tedy o zprávy ve formátu XML posílané protokolem HTTP. Komunikaci můžeme rozdělit na synchronní a asynchronní nebo na transientní (přechodná) a persistentní (vytrvalá).

Synchronní komunikace je používána například u vzdáleného volání procedur (RPC). Zde vždy klient pošle požadavek a čeká na odpověď od serveru. Teprve po obdržení výsledku může začít vykonávat jinou činnost. Oproti tomu asynchronní komunikace probíhá bez čekací doby. Klient pošle požadavek o vykonání metody na server a hned poté pokračuje v práci. Na příchod odpovědi ze serveru musí mít klient naimplementován oznamovací mechanismus (např. přerušení).

Druhé dělení určuje zabezpečení přenášení zpráv po síti. Persistentní komunikace garantuje doručení zprávy adresátovi. Naopak při transientní komunikaci se může zpráva ztratit při průchodu sítí.

1.1.5 Bezpečnost

Bezpečnost není řešena v rámci základních technologií WS. Právě proto vznikl standard organizace OASIS WS-Security, který řeší otázku zabezpečení komunikace. Existují dvě možnosti zabezpečení. Za prvé můžeme použít šifrovanou komunikaci na transportní vrstvě pomocí protokolu SSL a HTTPS. Druhý typ zabezpečení spočívá v použití standardu WS-Security, přesněji v šifrování a podepisování SOAP zpráv.

V případě použití protokolu HTTPS dochází k problému, kdy není možné rozdělit část webových služeb na šifrované a část na nešifrované. Jediná možnost je šifrovat všechny nebo žádné spojení. Další nevýhodou šifrování na transportní vrstvě je nemožnost použít prostředníka. Šifrování komunikace probíhá mezi koncovými uživateli a nikdo jiný nemá možnost rozumět přenášeným informacím. Další problém se objevuje s nemožností podepisování paketů šifrovaných pomocí protokolu HTTPS.

Pro šifrování a podepisování SOAP zpráv se vychází z použití standardů XML Encryption pro šifrování a XML Signature pro podepisování XML dokumentů. V SOAP zprávě přibývá element

„Security“, který definuje použité šifrování a podepisování. Tento způsob zabezpečení má oproti předchozímu protokolu HTTPS nevýhodu v nižší rychlosti zpracování.

1.1.6 Další specifikace

Mimo základní technologie SOAP, WSDL a UDDI (kapitola 1.1.3) byly vyvinuty další specifikace, které rozšiřují funkčnost WS. Zde je uvedeno jen několik z mnoha těchto specifikací. Specifikace jsou tvořeny mnoha firmami (např. Microsoft, IBM) nebo organizacemi (W3C, OASIS) a vždy jejich jméno začíná předponou „WS-“.

- WS-Security – definuje možnosti zabezpečení WS
- WS-Eventing – definuje jak se přihlásit k odebírání zpráv u WS nebo jak přijmout požadavek o odběr zpráv od jiné WS
- WS-Reliability – definuje spolehlivý přenos zpráv mezi WS
- WS-Addressing – definuje způsob popisu adresy odesilatele a příjemce uvnitř SOAP zprávy
- WS-Policy – popisuje vlastnosti WS, které nejde zapsat pomocí WSDL
- WS-Transfer – obdoba transportního protokolu HTTP pro přenos SOAP zpráv

1.2 SOAP

SOAP protokol je platformou nezávislý protokol sloužící k výměně informací mezi aplikacemi pomocí protokolu HTTP. Neboli protokol zprostředkovává komunikaci s WS. První verze 1.1 byla standardizována organizací W3C v roce 2000. Právě tato verze bude použita při komunikaci mezi serverem a klientem WS (více na [4]). Dnes existuje novější verze 1.2 z roku 2007.

Zprávy jsou tvořeny ve formátu XML splňující standard SOAP. Standard je kontrolován pomocí XML schématu, které definuje, jak musí SOAP zpráva vypadat. Pro komunikaci je možné použít různé protokoly, nejběžnější z nich je transportní protokol HTTP nebo jeho šifrovaná verze HTTPS. Při komunikaci se využívá operace POST. Do hlavičky HTTP je možné vkládat atribut „SOAPAction“ určující metodu, která má být zavolána. Další metoda jak určit, že se jedná o zprávu SOAP spočívá v nastavení atributu „Content-type“ na MIME typ „application/soap+xml“.

1.2.1 SOAP zpráva

Zpráva je podle standardu tvořena obálkou, v které můžeme najít hlavičku a tělo zprávy. Obálka je reprezentována kořenovým XML elementem „Envelope“, jeho potomkem je nepovinná hlavička (element „Header“) a tělo (element „Body“). Hlavička SOAP zpráv definuje specifické informace aplikace (častým případem použití může být autorizace). Pokud je hlavička přítomna, musí být první potomek elementu „Envelope“. Elementu „Body“ neboli tělo zprávy musí být vždy přítomno

v odesílané zprávě a za rodičovský prvek mít obálku (element „Envelope“). Potomci tohoto elementu obsahují aktuální SOAP zprávu (volanou metodu s parametry nebo její výsledek) určenou pro přijímající stranu.

Příklad klientské žádosti můžeme vidět na obrázku č. 3. Jedná se o zaslání požadavku na vykonání metody „RetrieveEventsForCalendar“, která stáhne všechny události pro určitý kalendář. Kalendář je určen pomocí parametru „calendarId“. V příkladu je vybrán výchozí kalendář. V hlavičce zprávy najdeme element „AuthHeader“, který slouží jako autorizace ke kalendářové službě.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthHeader xmlns="GCWebService">
      <userName>jan.novak</userName>
      <userPassword>novak00</userPassword>
    </AuthHeader>
  </soap:Header>
  <soap:Body>
    <RetrieveEventsForCalendar xmlns="GCWebService">
      <calendarId>default</calendarId>
    </RetrieveEventsForCalendar>
  </soap:Body>
</soap:Envelope>
```

Obrázek č. 3 – SOAP zpráva – požadavek odesílaný na WS

Odpověď na výše zaslaný dotaz je ukázána na obrázku č. 4. Zde už není nutná žádná autorizace, proto ze zprávy vypadla hlavička a zůstalo jen tělo. Opět hlavní část SOAP zprávy se nalézá v elementu „Body“. Další dva potomci naznačují, že se jedná o vrácený výsledek metody. Vrácenou hodnotou je XML dokument s kořenovým elementem „feed“ reprezentující seznam všech událostí daného kalendáře. Každá událost je tvořena elementem „entry“.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope .... >
  <soap:Body>
    <RetrieveEventsForCalendarResponse xmlns="GCWebService">
      <RetrieveEventsForCalendarResult>
        <feed>
          <entry>
            <title type="text">Koupit rohlíky</title>
            <author>
              <name>Jan Novák</name>
              <email>jan.novak@gmail.com</email>
            </author>
            ....
          </entry>
          ....
        </feed>
      </RetrieveEventsForCalendarResult>
    </RetrieveEventsForCalendarResponse>
  </soap:Body>
</soap:Envelope>
```

Obrázek č. 4 – SOAP zpráva – odpověď od WS

1.3 Web Services Description Language

WSDL (Web Services Description Language) patří mezi základní technologie, na kterých jsou WS postaveny. Jedná se o jazyk založený na XML, který popisuje WS a jak se k ní dá přistupovat. WSDL specifikuje operace (metody) WS a může také obsahovat její umístění (protokol a adresu). Dnes je WSDL ve verzi 2.0, která byla vydána organizací W3C jako doporučení „W3C Recommendation" (specifikace WSDL dokumentu dostupná na [5]).

WSDL dokumentu můžeme rozdělit na několik částí podle jeho hlavních elementů. Tyto prvky jsou potomky kořenového elementu „definitions". Mezi nejvýznamnější elementy patří prvek „portType", „message", „types" a „binding". V dokumentu WSDL se mohou vyskytovat i jiné elementy, například jde o rozšiřující prvky nebo o prvek, který spojuje více služeb do jednoho WSDL dokumentu. Dále můžeme dělit popis webové služby na abstraktní a konkrétní část. Do každé části připadají některé z hlavních prvků nebo jejich potomků.

Nejdříve si popíšeme hlavní prvky WSDL dokumentu, poté bude následovat abstraktní část a nakonec bude probrána konkrétní část.

1.3.1 Hlavní prvky WSDL

Zde bude popsáno, k čemu slouží a co obsahují hlavní prvky WSDL dokumentu.

- portType – Prvek „portType" je nejdůležitějším elementem celého dokumentu. Definuje totiž všechny operace s jejich popisem a názvem zpráv, které poskytuje daná WS . Operace mohou být typu: jednosměrné, žádost-odpověď, server zašle požadavek a čeká na odpověď nebo notifikace. Element „portType" plní stejnou funkci jako hlavičkový soubor nebo interface v tradičním programovacím jazyku.
- message – Prvek „message" definuje datové části určité operace. V tradičním programovacím jazyce přirovnáme tento element k parametrům funkce.
- types – Prvek „types" definuje datové typy použité ve WS. Pro jejich popis je použit syntax XML Schema.
- binding – Prvek „binding" definuje formát zprávy a protokoly, kterými je možné volat WS.

1.3.2 Abstraktní část

Abstraktní část WSDL dokumentu popisuje obecné vlastnosti a rozhraní WS. Nenajdeme zde žádné odkazy na použitou technologii a proto při změně platformy, zůstává tato část nezměněna. Patří sem elementy „operation", „message" a „interface". Operation obsahuje potomky „input" a „output" specifikující typ operace a příchozí, odchozí parametry. Tyto parametry jsou poté znovu definovány pomocí prvků „message". Element „message" je popsán v kapitole 1.3.1. Interface vyjadřuje abstraktní rozhraní WS.

1.3.3 Konkrétní část

V konkrétní části najdeme prvky, které se vážou na implementaci WS. Jedná se o elementy „binding“, „service“ a „endPoint“. Binding specifikuje konkrétní transportní protokol pro přenášení dat. Service definuje kolekci endpointů. Endpoint reprezentuje odkaz na instanci WS.

1.3.4 Příklad WSDL dokumentu

```
<wsdl:definitions name="nmtoken"? targetNamespace="uri"?>
    <import namespace="uri" location="uri"/>*
    <wsdl:documentation .... /> ?
    <wsdl:types> ?
        <wsdl:documentation .... />?
        <xsd:schema .... />*
    </wsdl:types>
    <wsdl:message name="nmtoken"> *
        <wsdl:documentation .... />?
        <part name="nmtoken" element="qname"? type="qname"?/> *
    </wsdl:message>
    <wsdl:portType name="nmtoken">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output name="nmtoken"? message="qname">?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken" message="qname"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="nmtoken" type="qname">*
        <wsdl:documentation .... />?
        <wsdl:operation name="nmtoken">*
           <wsdl:documentation .... /> ?
           <wsdl:input> ?
               <wsdl:documentation .... /> ?
           </wsdl:input>
           <wsdl:output> ?
               <wsdl:documentation .... /> ?
           </wsdl:output>
           <wsdl:fault name="nmtoken"> *
               <wsdl:documentation .... /> ?
           </wsdl:fault>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="nmtoken"> *
        <wsdl:documentation .... />?
        <wsdl:port name="nmtoken" binding="qname"> *
           <wsdl:documentation .... /> ?
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Obrázek č. 5 – příklad WSDL dokumentu (převzato z [5])

1.4 Universal Description Discovery and Integration

UDDI (Universal Description Discovery and Integration) je platformou nezávislý registr založený na XML. UDDI má sloužit jako celosvětový internetový rejstřík obchodních služeb. Technologie byla vyvíjena organizací OASIS, která jí v roce 2005 stanovila jako standard. UDDI vytváří veřejnou databázi všech služeb, do které nabízí možnost přidávat nové nebo v ní vyhledávat stávající WS. Registr sám o sobě také funguje jako WS a je možné s ním komunikovat pomocí protokolu SOAP.

Protože je UDDI katalog založen na veřejném přístupu, objevily se hned dva velké problémy spjaté s jeho chodem. Prvním problémem byla nefunkčnost nebo neplatnost většiny záznamů. Druhý problém je spojen s důvěryhodností WS. Pokud už se našly nějaké platné záznamy, nebylo možné ověřit jejich pravdivost. Kvůli těmto nedostatkům se přestaly UDDI rejstříky používat v internetovém prostředí a dnes je můžeme najít nanejvýš uvnitř firemního prostředí.

2 Google Calendar

Google Calendar (dále jen GC) slouží jako vždy přístupný kalendář s vysokou funkčností. Služba nabízí širokou správu kalendářů jak vlastních, tak veřejných. Umožňuje přidávat celodenní, časově omezené, opakující se události. K těmto událostem je poté možné přidat několik typů upozornění. Služba byla vytvořena společností Google a je funkční od dubna roku 2006. Aby ji bylo možné používat, je nutné založit si zdarma účet Google Account.

Nejdříve se podíváme na veškeré možnosti, které tato webová aplikace nabízí. Poté bude ukázáno webové rozhraní této služby. Dále bude probrána možnost vývoje vlastních aplikací propojených s GC. Nakonec bude uvedeno propojení GC s ostatními aplikacemi.

2.1 Možnosti služby Google Calendar

V této kapitole probereme veškerou funkčnost, kterou nám nabízí aplikace GC. Nejdříve se podíváme na možnosti ověřování uživatelů. Poté přijde podkapitola o kalendářích, dále si budeme moci přečíst o událostech a nakonec si povíme o upozorněních.

2.1.1 Ověření uživatele

Pro přihlášení k GC službě nám slouží tři možné typy ověřování. Prvním typem je tzv. AuthSub proxy ověřování, které je využíváno u webových aplikací, kdy aplikace nepotřebuje znát uživatelovo jméno a heslo. V tomto případě se uživatel přihlásí a aplikaci se přidělí token, přes který může komunikovat s uživatelským účtem. Dalším typem je přihlašování pomocí uživatelského jména a hesla. Posledním typem je použití tzv. Magic cookie ověřování. Každý kalendář má vygenerován svůj řetězec „Magic cookie“, pomocí kterého je možné stáhnout celý kalendář. Toto ověřování slouží pouze ke čtení GC.

2.1.2 Kalendáře

Při prvním spuštění aplikace je ve výchozím nastavení vytvořen primární prázdný kalendář. U každého kalendáře si můžeme nastavit jeho jméno, popis, umístění a časové pásmo. Dále můžeme zvolit, jestli má být kalendář veřejný nebo privátní. Publikovat můžeme kalendář dvěma způsoby. První umožní ostatním vidět všechny události. Druhá možnost zobrazí ostatním jen volný a obsazený čas. Nakonec můžeme přidávat určité uživatele, kteří budou mít speciální práva pro zacházení s tímto kalendářem. Uživatelům lze přiřadit jedno z čtyř možných práv. Nejvyšší právo umožňuje dělat změny a nastavovat sdílení. Nižší právo umožňuje jen dělat změny v událostech. Ještě nižší povoluje pouze vidět všechny události. Nejnižší ukazuje jen volný a obsazený čas. Poslední nastavení se týká webového rozhraní. Jedná se o barvu kalendáře, kterou budou vyplněny všechny události. Dále je

možné nastavit vlastnost skrytí nebo zobrazení kalendáře. Nakonec je možnost nastavit, jestli bude kalendář zatržený (vybraný) nebo ne.

Aplikace nabízí možnost mít hned několik kalendářů zároveň. Kalendáře dělíme na vlastní a veřejné. Vlastní kalendáře je možné vytvářet, upravovat a mazat. Veřejné je možné přidat k odebírání, omezeně nastavovat a odebrat. Možnost nastavování kalendáře závisí na právech, která byla uživatelům přiřazena.

2.1.3 Události

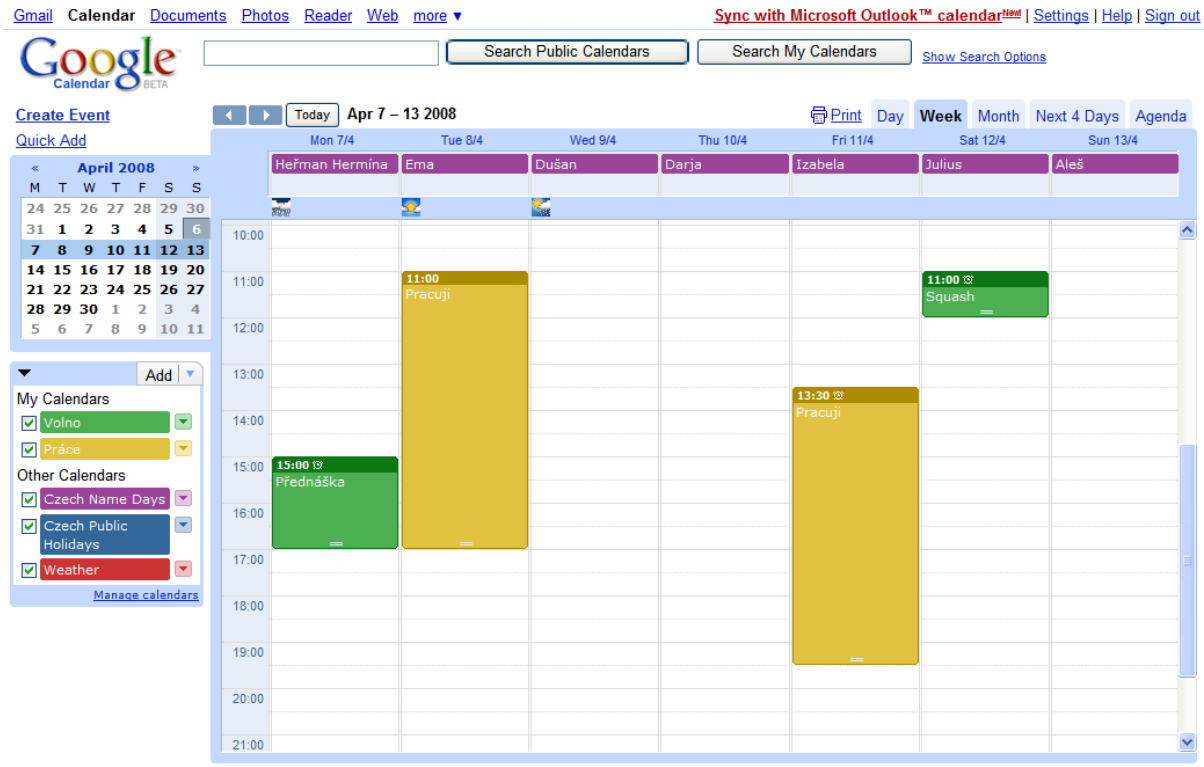
Každá událost nabízí nastavení předmětu (jména), popisu, umístění a data. Datum nastavujeme na celodenní nebo časově omezené. Dále je možné nařídit opakování po určitých časových intervalech (den, týden, měsíc,…). Poté můžeme určit, jestli je událost soukromá nebo veřejná nebo jestli se má zobrazovat jako volný nebo obsazený čas. Nakonec nám událost nabízí přidat upomínky k upozornění. Události poskytují stejné operace jako kalendáře, tedy vytváření, úpravu a mazání.

2.1.4 Upomínky

Upomínky je možné vytvářet u všech událostí. Existují tři typy upozornění. Prvním typem je otevření vyskakovacího okna s událostí (funguje jen přes webové rozhraní). Další typ upozorňuje pomocí emailu. Poslední typ posílá na telefon SMS zprávu s událostí. Tyto typy lze volně kombinovat.

2.2 Webové rozhraní

Webové rozhraní můžeme vidět na obrázku č. 5. V horní části najdeme hledací dialog, který umožňuje vyhledávat veřejné kalendáře nebo události ve vlastních kalendářích. Nalevo máme tlačítka na vytvoření událostí. Pod těmito ovládacími prvky se nachází kalendář se zobrazeným datem. Pod kalendářem je umístěn panel s vlastními kalendáři a s kalendáři přihlášenými k odebírání. Zde můžeme měnit všechny jejich vlastnosti. Pomocí tlačítka „Add“ přidáváme kalendáře. V spodní liště panelu se nachází odkaz, pomocí kterého lze kalendáře schovat nebo smazat. Zbytek obrazovky tvoří vybraný pohled s událostmi (v našem případě se jedná o týdenní zobrazení). Zobrazení lze zvolit v pravé horní části na denní, týdenní, měsíční, čtyřdenní a agendu, která vypisuje seznam po sobě jdoucích událostí.



Obrázek č. 5 – Webové rozhraní Google Calendar

2.3 Google API

Společnost Google nabízí vždy detailní popis aplikačních rozhraní svých aplikací. Webová aplikace GC není výjimkou a proto i Google Calendar API [15] s přesnou charakteristikou použití najdeme na webových stránkách společnosti. Pro vývojáře jsou vytvořeny příručky s příklady usnadňující vývoj nových aplikací. Google Calendar API nabízí veškerou funkčnost GC.

2.4 Propojení s ostatními aplikacemi

Aplikace GC nabízí možnost importování událostí ze souboru iCal nebo CSV (soubor vyexportovaný z aplikace MS Outlook) do vybraného GC kalendáře. Nově přibyla možnost synchronizace s aplikací Microsoft Outlook.

Další vlastností GC je schopnost vygenerovat HTML kód kalendáře a vložit ho do vlastních webových stránek. K tomuto účelu byl vytvořen speciální nástroj „Embeddable Calendar Helper Tool“, kde si uživatel za pár kliknutí vytvoří výsledný kód.

3 Hypertext Transfer Protokol

Protokol HTTP (HyperText Transfer Protokol) je internetový protokol vytvořený pro přenos hypertextových dokumentů. Protokol funguje na principu dotaz/odpověď. V praxi se používají dvě verze tohoto protokolu. První verze 1.0 vznikla jako první revize v květnu roku 1996. Aktuální verze 1.1 byla vytvořena v červnu roku 1999. Komunikace probíhá na protokolu TCP, ve většině případů se jedná o port 80. Přenos je vždy iniciován klientem, který zašle na server svoji žádost v podobě HTTP hlavičky (někdy může žádost obsahovat i tělo). Po vyřízení žádosti server odpovídá HTTP hlavičkou a tělem se zprávou (více informací na [13]).

Nejdříve si popíšeme hlavičky žádosti a odpovědí. Poté si ukážeme některé metody HTTP protokolu a nakonec probereme zabezpečení tohoto protokolu.

3.1 Hlavičky HTTP

Příklad klientské žádosti zaslané na server si můžeme prohlédnout na obrázku č. 6. Při navazování spojení se serverem je vždy nutné zadat metodu a verzi HTTP. Dále je nutné určit URI adresu serveru, na kterou se klient připojí. Do hlavičky můžeme zadat další informace. Na obrázku č. 6 vidíme v hlavičce parametr Host určující server, User-Agent popisující klienta (aplikaci, z které posíláme požadavek) a Accept-Charset stanovující znakovou sadu, které klient rozumí. Hlavička musí být oddělena prázdným řádkem.

```
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
User-Agent: Mozilla/5.0 Gecko/20040803 Firefox/0.9.3
Accept-Charset: UTF-8,*
--Prazdný řádek--
```

Obrázek č. 6 – HTTP hlavička – požadavek (převzato z [13])

Po obdržení zprávy od klienta server zpracuje požadavek a pošle odpověď. Odpověď je typická HTTP hlavičkou a tělem zprávy (příklad odpovědi je vidět na obrázku č. 7). Hlavička obsahuje verzi protokolu, kód a text odpovědi. Dále v hlavičce najdeme ostatní informace jako čas odeslání, typ serveru atd. Opět je v zprávě hlavička oddělena od těla prázdným řádkem.

```
HTTP/1.0 200 OK
Date: Fri, 15 Oct 2004 08:20:25 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.8
X-Powered-By: PHP/4.3.8
Vary: Accept-Encoding,Cookie
Cache-Control: private, s-maxage=0, max-age=0, must-revalidate
Content-Language: cs
Content-Type: text/html; charset=utf-8
--Prazdný řádek--
<HTML>...
```

Obrázek č. 7 – HTTP hlavička – odpověď (převzato z [13])

Pokud nastane nějaká chyba, je vrácen chybový kód v hlavičce odpovědi. Na obrázku č. 7 jsme se setkali s kódem 200, který značí, že vše proběhlo v pořádku. Podle první číslice můžeme poznat, o jaký typ odpovědi se jedná (1xx - informativní, 2xx - úspěch, 3xx - přesměrování, 4xx - chyba u klienta nebo 5xx - chyba na serveru).

3.2 Metody HTTP

Metoda reprezentuje druh služby, o kterou žádá klient. Některé metody nemusí server podporovat a poté vrací chybový kód.

- GET – nejčastěji používaná metoda, která žádá o zaslání dokumentu specifikovaného pomocí URL
- POST – metoda posílá informace od klienta na server v těle zprávy, používá se například při posílání rozsáhlých webových formulářů, nahrávání souborů na server, atd. Rozsah přenášených dat není omezen.
- PUT – metoda posílá žádost o uložení posílaných dat na serveru
- DELETE – metoda žádá o zrušení dokumentu (určen pomocí URL) na serveru.
- HEAD – podobná metoda jako GET, ale odpověď obhajuje pouze hlavičku
- OPTIONS – metoda zjišťuje informace o dané URL (serveru nebo dokumentu)
- TRACE – metoda se používá k testování, vrací cestu celého dotazu

3.3 Zabezpečení HTTP

HTTP protokol je přenášen po síti v otevřeném formátu. Pokud ho potřebujeme nějak zabezpečit, můžeme využít protokol HTTPS. Jedná se o nadstavbu protokolu HTTP s šifrováním dat pomocí SSL nebo TLS. Komunikace opět funguje na protokolu TCP a nejčastěji na portu 443.

4 Extensible Markup Language

XML (eXtensible Markup Language) slouží jako obecný značkovací jazyk. Byl vytvořen a standardizován organizací W3C a je navržen pro přenos a uchování dat. XML nedefinuje strukturu dokumentu, ta je vytvořena při vývoji aplikace podle potřeby. Novou strukturu je možné definovat pomocí XML schématu, které slouží jako metadata XML. Dále je možné stylovat XML dokument pomocí XSL a vytvořit tak HTML dokument nebo jiné formáty vhodné pro prohlížeč. XML můžeme i transformovat například do jiného dokumentu nebo do jiného XML pomocí XSLT. XML je využíváno v XHTML (nástupce HTML), v SOAP (komunikace mezi WS), v RSS (novinky na internetových stránkách) a v mnoha dalších aplikacích a protokolech (více na [14]).

V této kapitole si ukážeme příklad XML dokumentu a popíšeme si jeho strukturu. Dále si povíme o XML namespace a jejich funkci v XML. Poté se zaměříme na definici XML, přesněji na DTD a XML schéma. Nakonec si uvedeme možná rozšíření XML.

4.1 XML dokument

Při tvorbě dokumentu je třeba dodržovat striktní syntaxi dokumentu. V porovnání se syntaxí HTML, je syntaxe přísnější. XML dokument může být deklarován na prvním řádku textu, uvádí se v něm verze XML. Spolu s verzí může obsahovat kódování textu nebo externí závislosti. Dále následuje kořenový element. XML dokument musí obsahovat jediný kořenový prvek. Následuje text dokumentu, popřípadě další elementy (potomci kořenového elementu). Poslední řádek obsahuje koncový kořenový element. Každý element může obsahovat atributy, text mezi počátečním a koncovým elementem nebo další potomky. Příklad takového XML dokumentu můžeme zhlédnout na obrázku č. 8.

```
<?xml version="1.0" encoding="UTF-8"?>
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
   <title>Basic bread</title>
   <ingredient amount="3" unit="cups">Flour</ingredient>
   <ingredient amount="0.25" unit="ounce">Yeast</ingredient>
   <ingredient amount="1.5" unit="cups" state="warm">Water</ingredient>
   <ingredient amount="1" unit="teaspoon">Salt</ingredient>
   <instructions>
     <step>Mix all ingredients together.</step>
     <step>Knead thoroughly.</step>
     <step>Cover with a cloth, and leave for one hour in warm room.</step>
     <step>Knead again.</step>
     <step>Place in a bread baking tin.</step>
     <step>Cover with a cloth, and leave for one hour in warm room.</step>
     <step>Bake in the oven at 350(degrees)F for 30 minutes.</step>
   </instructions>
 </recipe>
```

Obrázek č. 8 – XML dokumentu (převzato z [14])

4.2 DTD a XML schéma

XML dokumenty můžou být definovány pomocí DTD (Document Type Definition) nebo XML schématu. DTD a XML schéma slouží jako metadata výsledných XML dokumentů a používají se tedy i k jejich validaci. Existují nástroje, které dokážou porovnat XML dokument s jeho definicí určenou DTD nebo schématem. Dále se DTD a XML schéma používá například v XML editorech, kde uživateli aplikace nabízí povolené elementy.

DTD představuje nejstarší schéma pro XML dokumenty, které vychází z SGML. DTD nepodporuje nové vlastnosti XML (např.: namespace, datové typy, atd.). Popis XML není zapsán v XML syntaxi a je použit popis zděděný z SGML. V dnešní době se DTD stále používá, to je způsobeno hlavně díky lehké čitelnosti a zápisu kódu.

Novější XML schéma je popsáno organizací W3C a mělo by sloužit jako nástupce starého DTD. Zde je definice dokumentu tvořena v XML syntaxi a umožňuje popisovat více vlastností než DTD. Díky tomu XML schéma může popisovat jména, pořadí a datové typy elementů a jejich atributů.

4.3 XML namespace

Při tvorbě XML dokumentu se můžeme setkat s tím, že budeme chtít použít více XML schémat najednou. Právě za tímto účelem vznikly namespace, které oddělují definice schémat, a tedy nemůže dojít k nejednoznačnosti značek. Schémata jsou v dokumentu definována pomocí formátu „xmlns:prefix="URI"“, kde na URI adrese najdeme dané schéma. Definice namespace se uvádí jako atribut elementu, pro který má platit dané schéma. Pokud chceme vytvořit element, který vychází z daného schématu, musíme uvést jeho prefix před značkou. Formát elementu vypadá poté takto „<prefix:značka>“. Pokud neuvedeme žádný prefix při definici namespace, je definice značek použita na všechny elementy, které taky nemají definovaný žádny prefix.

4.4 XML Rozšíření

Jazyk XML obsahuje několik rozšíření, která zajišťují například jednodušší průchod dokumentem (XPath), zabezpečení dokumentu při přenosu sítí (XML Signature, XML Encryption) a jiné (XML Namespace…). XPath slouží k nalezení určité části dokumentu. Může se jednat o element, atribut, komentář, text, data atd. XML Signature definuje, jak vytvořit digitální podpis obsahu XML dokumentu. XML Encryption definuje, jak se dá obsah dokumentu šifrovat.

5 Návrh řešení

V této kapitole se seznámíme s požadavky na výslednou serverovou aplikaci vytvářející rozhraní pomocí webových služeb mezi webovou (Google Calendar) a klientskou aplikací. Dále si popíšeme, jak má vypadat klientská část aplikace zaměřená na demonstraci komunikace webových služeb.

Poté si ukážeme možný návrh serverové aplikace, která bude vytvářet standardizované rozhraní pro přístup ke všem metodám webové aplikace GC. Bude tedy sloužit jako prostředník mezi komunikací klienta a serveru GC.

Předposlední podkapitola nám popíše klientskou aplikaci vytvořenou za účelem ukázkové aplikace webových služeb.

Nakonec bude vysvětlena komunikace mezi všemi prvky systému (GC server, server WS, klientská aplikace a registr UDDI). Ukážeme si postup připojení WS do katalogu a následnou komunikaci s klientem.

5.1 Neformální specifikace

Základním úkolem je vytvořit aplikaci na bázi webových služeb, která by sloužila jako rozhraní pro službu Google Calendar. Dále je potřeba vyvinout program, který by dokázal komunikovat s tímto rozhraním. Díky těmto dvěma odlišným částem budeme výslednou aplikaci popisovat odděleně, nejdříve serverovou a poté klientskou část.

Jedním z nejdůležitějších požadavků na serverovou aplikaci je vytvoření standardizované webové služby, která bude poskytovat popis veškerých metod ve formátu WSDL a bude komunikovat prostřednictvím protokolu SOAP. Na serveru je tedy třeba implementovat všechny funkce na vytvoření, úpravu a mazání kalendářů, událostí, upomínek a práv uživatelů na přístup k těmto údajům. Server musí rozumět všem SOAP žádostem vytvořeným podle WSDL popisu a umět zpracovat více těchto žádostí najednou. Dalším neméně důležitým předpokladem je komunikace se serverem nabízejícím službu GC. Mimo tyto hlavní vlastnosti je třeba implementovat zabezpečenou komunikaci s GC serverem a s klientem, aby nemohlo být odposlechnuto heslo k účtu GC služby nebo nebyly přečteny důvěrné informace obsaženy v kalendářích. Aby mohl začít klient vůbec komunikovat s WS, je třeba ji vyhledat v katalogu UDDI. Ještě před jejím hledáním se musí tato služba zaregistrovat do katalogu UDDI. K tomuto úkonu je tedy nutné implementovat metody na přidání služby do rejstříku WS nebo použít stávající metody UDDI.

Klientská aplikace má být navržena s ohledem na výukové účely webových služeb, právě proto většina funkčnosti bude orientována na parsování WSDL dokumentu a komunikaci pomocí SOAP zpráv mezi klientem a WS. Dále klient musí umět vyhledat službu v předem určeném UDDI katalogu a stáhnout si její WSDL popis. Dále by měla aplikace umožnit vlastní psaní SOAP zpráv s následnou

validací a upozorněním nad chybami v kódu. Díky standardizovanému rozhraní webových služeb musí být možné použít klientskou část na komunikaci s jakoukoli WS.

5.2 Webová služba Google Calendar

Je třeba vytvořit konkurentní server, který bude rozumět HTTP požadavkům a bude na ně umět odpovídat pomocí HTTP odpovědí. Dále je nutné napojit se na rozhraní GC a vytvoříme tak přístup k všem ovládacím prvkům služby. Poté nadefinujeme metody pro uživatele WS, přes které bude spravovat celou službu. Ke každé metodě vytvoříme příslušné SOAP zprávy. U každé metody je potřebná autorizace k příslušnému účtu s kalendáři. Ta může být řešena pomocí SOAP hlaviček u všech metod.

Jako implementace serveru může být použita platforma .NET s jazykem C#, která automaticky generuje příslušné SOAP zprávy a k nim i popis ve WSDL. Překompilovaný kód bohužel potřebuje server, který se dají spouštět stránky ASP.NET. Jedním z takovýchto serveru je aplikace IIS (Internet Information Service) od firmy Microsoft. V této aplikaci můžeme nastavit i použití protokolu HTTPS a tím úspěšně zabezpečit celou aplikaci.

5.3 Klientská aplikace

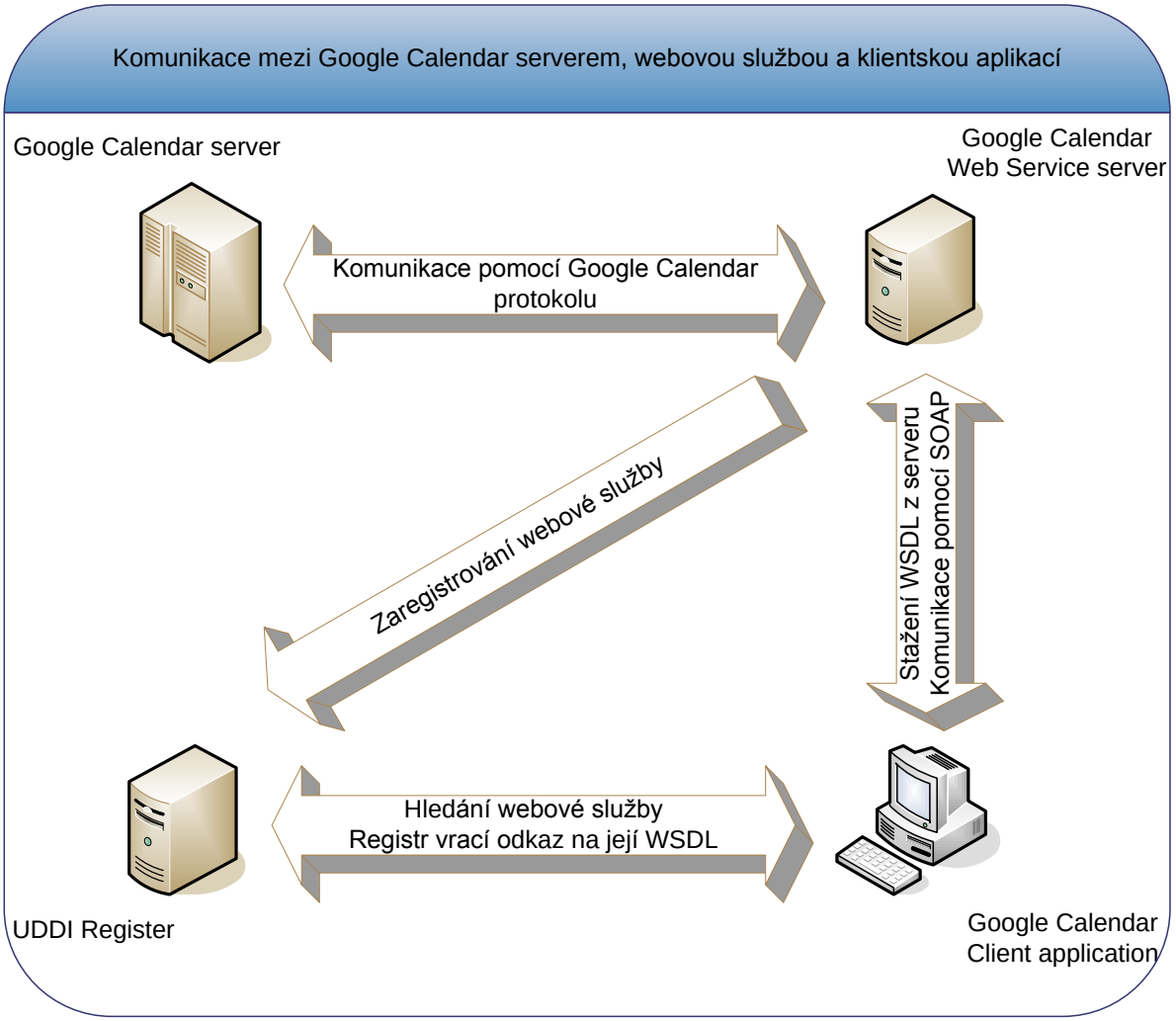
Klientská aplikace může být naprogramována například jako desktopová aplikace, která bude umět komunikovat s registrem UDDI a se serverem WS. Na uživatelův pokyn kontaktuje program rejstřík a zkusí najít hledanou službu. Po nalezení popisu aplikace automaticky stáhne WSDL soubor a rozparsuje ho. Uživateli budou následně nabídnuty všechny metody, kterým WS rozumí a umí je vykonat. Po zvolení metody si uživatel může napsat vlastní SOAP zprávu podle WSDL popisu nebo použít předdefinovanou zprávu, do které doplní jen potřebné data. Posledním krokem je kontrola struktury zprávy, její odesílání na server a obdržení odpovědi.

5.4 Komunikace mezi aplikacemi

Výslednou komunikaci mezi všemi aktéry, kteří se podílí na funkčnosti celé aplikace, můžeme vidět na obrázku č. 9. Jsou zde vidět 4 hlavní části, které mezi s sebou komunikují. První je Google Calendar server, na kterém běží aplikace Google Calendar. Další je server s webovou službou nabízející ovládání GC. Tato část je propojena se serverem GC pomocí Google Calendar protokolu a slouží k posílání jakýchkoli požadavků/odpovědí na/z server GC. Dále je WS napojena na všechny ostatní účastníky komunikace. Jedním z nich je klientská aplikace, která komunikuje s WS pomocí SOAP zpráv a ovládá tak službu GC. Do této komunikace také patří stažení popisu rozhraní WSDL klientskou aplikací. Klient je také propojen s posledním prvkem komunikace, kterým je UDDI registr.

Propojení představuje hledání WS a případnou odpověď s odkazem na službu. Komunikace s rejstříkem služeb je využívána i webovou službou GC, která jí používá k vlastní registraci mezi ostatní WS.

Nyní si popíšeme postup, jakým probíhá komunikace mezi těmito čtyřmi členy, při vytvoření a používání WS. Na začátku vzniká propojení mezi GC webovou službou a GC serverem. Dále po vzniku plnohodnotné WS se služba registruje do katalogu služeb v registru UDDI. Po tomto kroku je vše připraveno k použití a čeká se na klienta, až zahájí komunikaci. Když je klient připraven, vyhledá si GC službu v UDDI adresáři a obdrží na ni odkaz. Poté kontaktuje WS na získaném odkazu a stáhne si její WSDL popis. Poslední částí je komunikace s WS přes SOAP zprávy vytvořené podle specifikace WSDL.



Obrázek č. 9 – Návrh komunikace mezi servery a klientem

6 Implementace

Šestá kapitola popisuje vytvoření celé práce. Jedná se tedy o popis implementace klienta a webové služby. Nejdříve se podíváme na programovací jazyk, který byl použít při implementaci. S tímto jazykem bude spjata i platforma, na které jsou napsané programy vykonávány.

Poté si detailně vysvětlíme implementaci webové služby Google Calendar. Ukážeme si vytvořenou WS metodu s jejími SOAP zprávami a WSDL popisem. Bude možné vidět webové rozhraní WS.

Nakonec si představíme klientskou aplikaci. Její popis bude rozdělen na dvě části. První část se zabývá vyhledáním a nalezením WSDL popisu. Část druhá popisuje následnou výměnu informací s WS pomocí SOAP zpráv.

6.1 C# .NET

Jako programovací jazyk pro vyhotovení celé práce byl použit jazyk C#. Jedná se o objektový jazyk vytvořený firmou Microsoft. Pro chod všech aplikací napsaných v tomto jazyce je nezbytné mít nainstalovanou platformu .NET, na které aplikace běží. Obě aplikace (klient i server) využívají platformu .NET ve verzi 2.0.

Při tvorbě webové služby byly využity výhody platformy .NET. Jednou z jejích vlastností je automatické generování SOAP zpráv a WSDL popisu podle nadefinovaných metod. Dále vytváří webové rozhraní pro přehled všech metod WS s ukázkou SOAP zpráv. Díky objektově orientovanému přístupu vznikl lépe čitelný a efektivnější kód. Jednou z nevýhod platformy je nutnost použití serveru ASP.NET (př.:IIS) pro chod webové služby.

6.2 Webová služba Google Calendar

Tato kapitola vysvětluje implementaci celé serverové části. Serverová aplikace byla vytvořena podle návrhu z předchozí 5. kapitoly. Kapitola je rozdělena na část implementace a část webové rozhraní. První podkapitola popisuje celou funkčnost a implementaci aplikace. Druhá ukazuje webové rozhraní služby a její použití.

6.2.1 Implementace

Webová služba obsahuje dvě hlavní třídy (Calendar, Service), které se starají o bezchybný chod celé serverové aplikace. Třída Calendar je statická třída, která zprostředkovává propojení se službou GC. Jedná se tedy o zasílání dotazů na server GC a následné přijímání odpovědí. Všechna probíhající komunikace je synchronní. Třída obsahuje vlastní metodu pro každou metodu WS. Po zavolání

metody WS se předají parametry příslušné metodě z třídy Calendar, která požadavek vykoná a vrátí odpověď. Třída Service vytváří všechny metody WS a stará se o kontrolu zaslaných parametrů a o zpracování případných chyb. Dále slouží k formátování zaslaného výsledku klientovi, kdy výsledek z každé metody třídy Calendar serializuje na XML dokument a pošle klientovi. Kromě metod WS jsou zde pomocné funkce pro práci s XML a pro zpracování chyb.

Každý klient volající jakoukoli metodu WS musí být nějakým způsoben autorizován u služby GC. Tato autorizace je zprostředkována pomocí použití hlaviček SOAP zpráv. Proto v každém volaném SOAP požadavku musí být uvedena SOAP hlavička se jménem a heslem k službě GC.

Celá komunikace probíhá v následujícím pořadí. Nejdříve pošle klient požadavek na vykonání metody WS. Po jejím obdržení server, přesněji objekt Service, kontroluje přítomnost SOAP hlavičky a korektnost formátu zaslaných parametrů. Dále je požadavek předán statické metodě z třídy Calendar, která se žádost pokusí vykonat. Výsledek nebo případná chyba je vrácena zpět metodě objektu Service, kde se výsledek upraví na XML dokument nebo vzniklá chyba zpracuje a zašle zpět klientovi.

Kromě těchto dvou nejvýznamnějších tříd obsahuje serverová část také pomocné objekty na předávání dat a definici vlastních výjimek. Mimo aplikační kód zde také nalezneme datové soubory definující omezení u některých parametrů metod WS. Všechny datové soubory jsou uloženy ve formátu XML. První soubor „calendarColors.xml“ obsahuje jediné možné barvy kalendáře ve webovém rozhraní. Tyto barvy se zadávají jako parametry metody pro vytvoření nebo úpravu kalendáře. Dalším souborem je „eventReminderTimes.xml“, který určuje platné časové úseky pro nastavení upomínky. Jedná se o dobu, po které má nastat upozornění. Čas se nastavuje jako parametr při tvorbě upomínky u zvolené události. Poslední soubor „timezones.xml“ definuje validní časové pásma kalendáře. Opět se tato vlastnost nastavuje u metod vytváření nebo úpravy kalendáře.

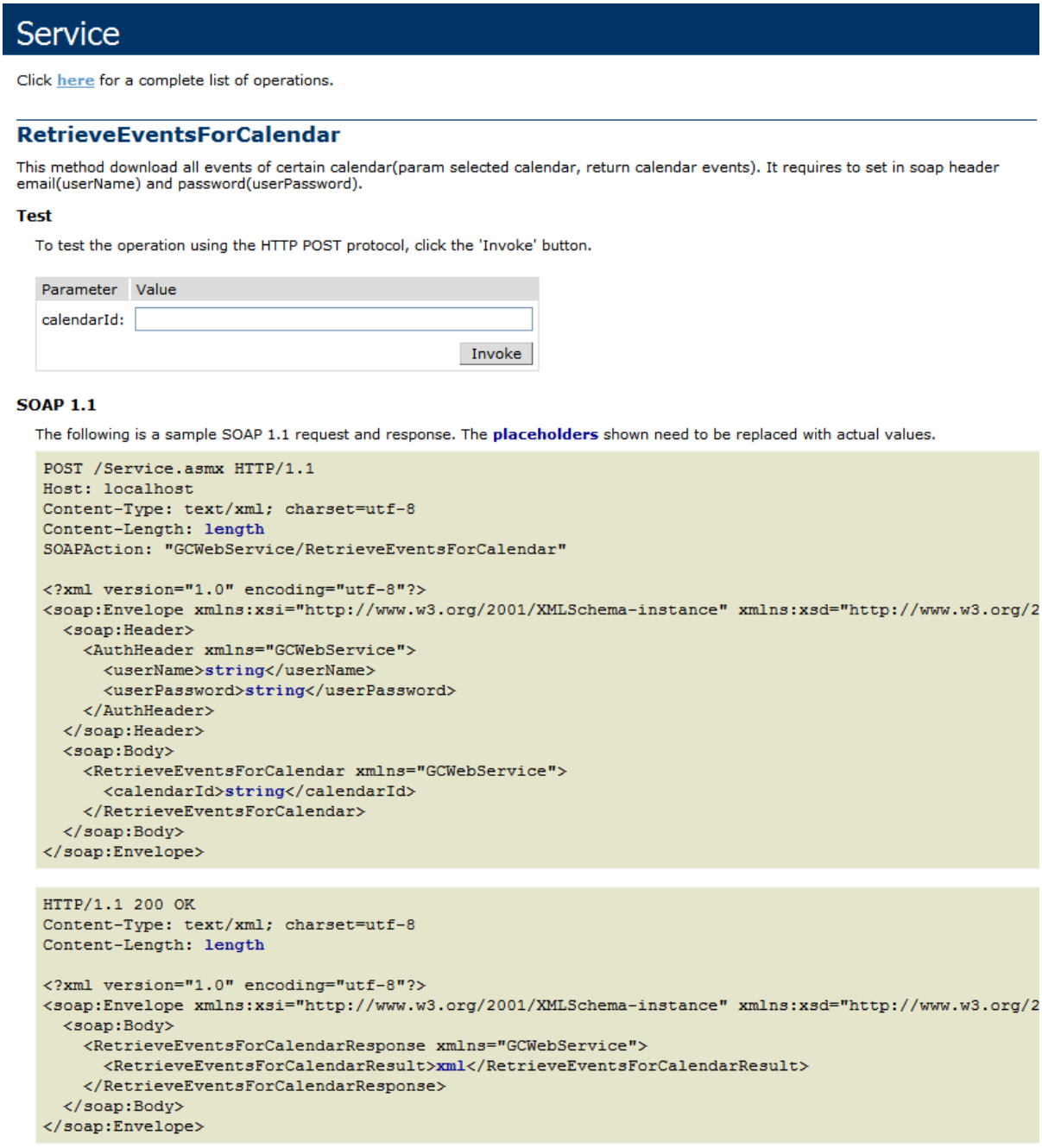
Při implementaci byly využity dynamické knihovny Google Data API, které umožňují jednoduché propojení s aplikací Google Calendar. Po přeložení serverové části vzniká další dynamická knihovna obsahující její celý kód. Kromě dynamických knihoven obsahuje výsledná webová služba také datové soubory (calendarColors.xml, eventReminderTimes.xml, timezones.xml), konfigurační soubor a soubor WS (Service.asmx) odkazující se na DLL knihovnu.

6.2.2 Webové rozhraní

Po spuštění webové služby je možné kromě zasílání SOAP požadavků a přijímaní jejich odpovědí prohlížet všechny metody WS na webovém rozhraní serveru. Platforma .NET automaticky vytváří webové stránky, kde si můžeme prohlédnout všechny metody a k nim patřičné SOAP požadavky a odpovědi. Dále je zde také odkaz na WSDL popis služby. Webové rozhraní se dá také použít i jako dobré ladící prostředí. Pokud máme službu spuštěnu na localhostu, můžeme jí zasílat požadavky

přímo z tohoto webového rozhraní pomocí metody POST. Bohužel do zasílané žádosti jsme schopni vyplnit pouze parametry těla SOAP zprávy.

Na následujícím obrázku č. 10 je možné vidět ukázku webového rozhraní zobrazující metodu RetrieveEvetsForCalendar. Tato metoda stahuje všechny události pro daný kalendář. V horní části obrazovky je vypsán název metody a její popis. Níže je umístěno testovací okno, které umožňuje poslat požadavek s touto metodou na server. Zbytek plochy tvoří ukázka SOAP zprávy verze 1.1. První hnědé okno obsahuje dotaz na WS včetně hlavičky HTTP protokolu. Spodní okno představuje odpověď na daný dotaz. Všechna modrá slova musí být vyměněny za pravé hodnoty.



Obrázek č. 10 – Webové rozhraní – Metoda stahující události ze zvoleného kalendáře

6.3 Klientská aplikace

V této kapitole je možné najít popis implementace klientské části projektu. Bude se tedy jednat o popis desktopové aplikace, která umí zpracovávat WSDL popis a následně komunikovat s příslušnou WS. Už podle specifikace můžeme aplikaci rozdělit na dvě hlavní části. Nejdříve si popíšeme první část zabývající se zpracováním WSDL. Poté bude probrána část zabývající se komunikací s WS pomocí SOAP zpráv.

6.3.1 WSDL parser

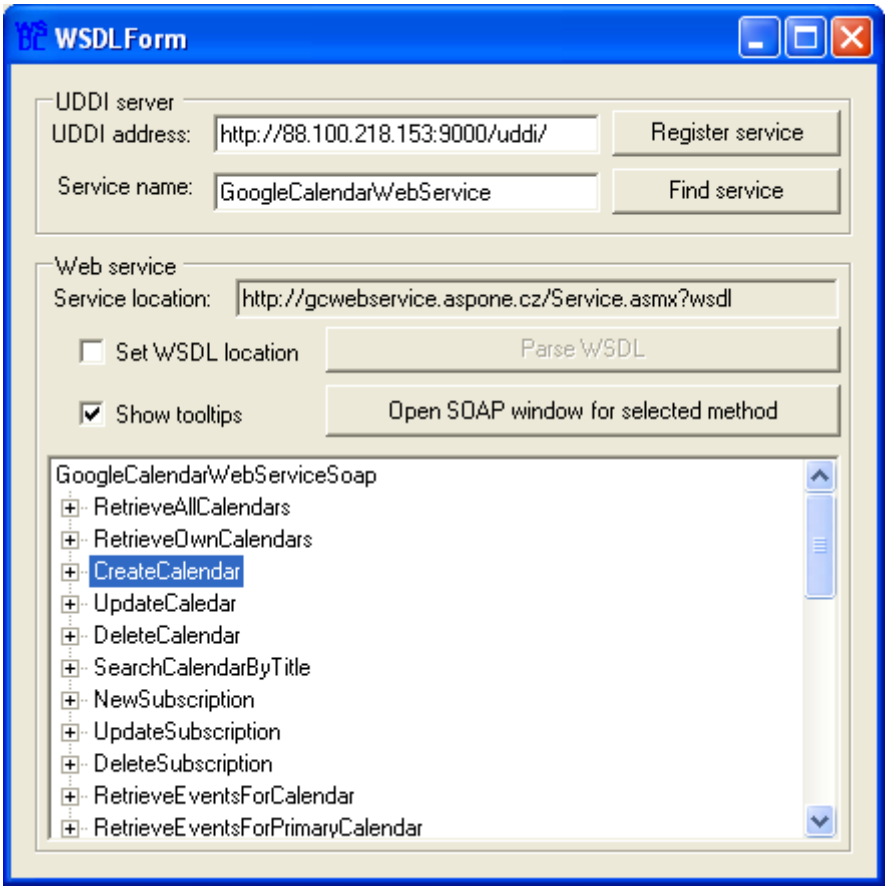
Dříve než je možné komunikovat s WS pomocí SOAP zpráv, musíme zjistit jaké metody WS nabízí a v jakém formátu jsou zasílány. Právě k tomuto účelu slouží úvodní okno aplikace, které umožňuje registraci, nalezení služby a zpracování jejího popisu. Toto okno je implementováno třídou WSDLForm. Kromě metod starajících se o chod grafického režimu okna, třída také obsahuje metody pro vyvolání SOAP okna a pro komunikaci s UDDI registrem.

Při registraci či hledání WS je kontaktován UDDI katalog. Protože katalog sám funguje jako WS, komunikace probíhá pomocí SOAP zpráv. Po stisku registračního tlačítka je otevřeno nové okno s příslušnou SOAP zprávou (kapitola 6.2.3). Toto okno načítá formát požadavku a odpovědi ze souboru „save_business.xml“ a adresa UDDI serveru z hlavního okna. Do SOAP žádosti vyplníme potřebné údaje a odešleme na server. K hledání WS jsou použity pole adresy UDDI registru a jméno WS. Po vyplnění těchto údajů a stisku tlačítka pro nalezení služby je kontaktován UDDI server. Formát zasílané zprávy najdeme v souborech „find_business.xml“ a „get_business.xml“. Pokud je služba nalezena, je nabídnuto dialogové okno s možností zpracování popisu služby. Jestliže nastane chyba nebo služba není zaregistrována v katalogu, objeví se příslušné chybové hlášení.

O parsování WSDL popisu se starají třídy ze jmenného prostoru GCClient.WSDL. Kde GCClient je jmenný prostor celé klientské aplikace. Tyto třídy byly převzaty z projektu „WSDL and Schema Parser“ [16] a následně upraveny. Po nalezení WS a stažení jejího popis, je vytvořen objekt třídy WSDLparser, který rozparsuje celý popis do stromu metod WS. Tyto metody můžeme procházet v okně aplikace. Po zvolení určité metody je možné otevřít nové okno určené pro komunikaci se službou pomocí SOAP zprávy vytvořené z této metody.

Na obrázku č. 11 je ukázána počáteční obrazovka aplikace s vyhledanou a načtenou webovou službou Google Calendar. Okno je rozděleno na části „UDDI server“ a „Web Service“. První část se stará o komunikaci s UDDI registrem a druhá část se zabývá WS a jejím popisem. V tomto případě se vyhledávala služba „GoogleCalendarWebService“ na UDDI serveru „88.100.218.153:9000“. Po jejím nalezení došlo k zpracování WSDL popisu této služby a následnému vypsání všech metod v podobě stromu do vnitřního okna části „Web service“. Pokud posuneme ukazatele myši nad jakýkoli prvek stromu, je ukázán jeho popis v podobě žlutého vyskakovacího okna. Po označení určité metody je možné otevřít nové okno pro komunikaci s WS pomocí tlačítka „Open SOAP Windows for selected

method“. Toto okno umožňuje zaslání označené metody webové službě ve formě SOAP dotazu. Zde byla vybrána metoda „CreateCalendar“ a její okno pro zasílání požadavků je vidět na obrázku č. 12 v následující kapitole 6.3.2.



Obrázek č. 11 Klientská aplikace – Úvodní okno

6.3.2 Komunikace pomocí SOAP

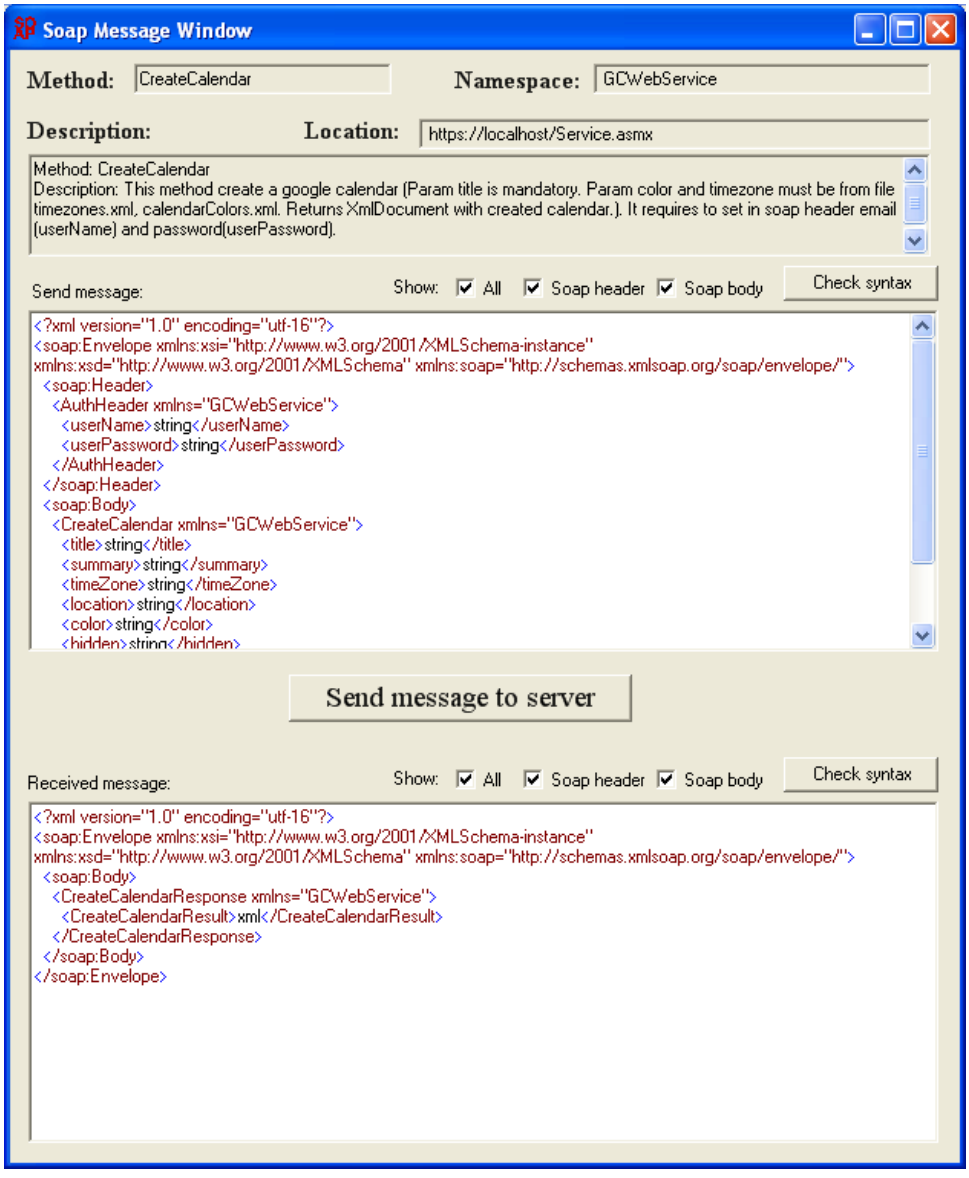
Druhá část nebo také druhé okno aplikace nabízí komunikaci se serverem pomocí SOAP zpráv. Umožňuje posílání zpráv na server WS a poté následné přijímání odpovědí. Okno je vždy vyvoláno pro určitou metodu WS a obsahuje tedy pokaždé specifické informace této metody. Můžeme zde najít jméno metody, její popis, namespace, adresu serveru a formát odesílané či přijímané zprávy.

Protože je aplikace určená pro výukové účely, je zde možnost napsat si vlastní SOAP žádost podle WSDL popisu. Tento požadavek je možné zkontrolovat pomocí XML schématu SOAP zprávy, tato kontrola je implicitně vykonána před odesláním požadavku. Pokud má zpráva chybnou syntaxi, je vyvoláno chybové hlášení a žádost se neodesílá. Chyba může být způsobený špatnou syntaxí SOAP zprávy nebo chybějícím elementem. Pokud ve zprávě chybí určitý prvek, je na něj uživatel upozorněn. O validaci XML zprávy se stará statická třída „XML“. Aplikace dále nabízí možnost dopsání celé nebo jen části SOAP zprávy. K tomuto účelu jsou v okně programu umístěny checkboxy určující, které elementy se mají doplnit. Jestliže zvolíme dopsání některé části, program ji

automaticky vyplní v okně zprávy. Tento úsek kódu obsahuje formát SOAP zprávy s datovými typy, které je třeba nahradit požadovanými informacemi.

Kromě zmíněné statické třídy XML je v aplikaci použita třída SoapForm, ColorTextBox a SoapMethods. Třída SoapForm vytváří celé okno se SOAP metodou a je tedy nejdůležitější třídou z této části aplikace. Třída ColorTextBox se používá jako komponenta pro psaní SOAP zpráv, která automaticky barevně zvýrazňuje text podle syntaxe XML. Poslední třída SoapMethods implementuje objekt, který nese všechny potřebné informace pro komunikaci se serverem WS. Dále tato třída obsahuje statické metody pro přidání SOAP hlavičky a těla.

Celé okno s metodou je vidět na následujícím obrázku č. 12. Obrázek zobrazuje SOAP metodu na vytvoření kalendáře. Ve vrchní části můžeme najít její základní informace. Pod těmito údaji se nachází část pro tvorbu požadavku. Uprostřed okna můžeme najít tlačítko na odesílání požadavku na server WS. Úplně vespod okna se nachází část pro příchozí zprávu.



Obrázek č. 12 – Klientská aplikace – Okno se SOAP metodou

7 Závěr

V této práci jsme se seznámili z technologií webových služeb a předvedli jsme si architekturu SOA, na které je aplikace založena. Uvedli jsme si komunikaci pomocí SOAP zpráv a jejich popis ve formátu WSDL. Poté byl vysvětlen pojem UDDI a jeho propojení s webovými službami v podobě registru uchovávajícího informace o webových službách. Nakonec byly uvedeny pojmy HTTP a XML, které jsou nedílnou součástí webových služeb, protože se podílejí na celé jejich funkčnosti. Z popisu a výsledné aplikace můžeme vidět univerzálnost webových služeb, jejich potenciál a široké uplatnění při komunikaci přes internet.

Prvotní zadání práce týkající se propojení webové služby se službou Google Calendar spočívalo jen v nabídnutí čtení stávajících kalendářů k určitému účtu. To znamenalo veliké omezení klientské aplikace, která by mohla jen číst a filtrovat aktuální události. Tento přístup byl nakonec rozšířen o celou správu služby Google Calendar. Nyní metody webové služby nabízejí veškerou funkčnost, která je dostupná z webového rozhraní Google Calendar. Je tedy možné pomocí zasílaných SOAP požadavků kalendáře, události, upomínky a práva vytvářet, upravovat a mazat. Možné omezení můžeme vidět v bezpečnostní stránce aplikace, kdy celá webová služba je zabezpečena pouze serverovou aplikací, na které je spuštěna. Pokud je použit nezabezpečený protokol HTTP, celá komunikace probíhá v otevřeném formátu. Jestliže někdo odchytí tyto přenášené informace, může si je bez větších problémů přečíst a zjistit tak jméno a heslo k příslušnému účtu Google Calendar.

Klientská část byla původně určena jen k demonstrační funkci výsledné serverové aplikace webové služby. Toto zadání bylo následně také upraveno na případné použití aplikace jako výukového prostředku webových služeb znázorňujícího komunikaci s UDDI registrem a webovou službou. Konečná podoba klientské aplikace umožňuje propojení s jinou bakalářskou prací zaměřenou na tvorbu UDDI registru. Komunikace s touto databází webových služeb spočívá v registraci nové WS nebo vyhledání stávající WS. Jedním z omezení klientské aplikace je použití jednoho vlákna na každé zobrazované okno. Tato implementace způsobuje chvilkové zatuhnutí okna například při větší latenci nebo výpadku sítě při komunikaci se servery. Další omezení najdeme v načítání popisu webové služby. Protože klientská aplikace využívá pouze jednoduchý parser na zpracování WSDL popisu, je nutné, aby všechny definované elementy byly právě v tomto jednom souboru.

Nyní si popíšeme možná rozšíření webové služby a klientské aplikace. Většina těchto rozšíření bude zaměřena na odstranění dosavadních nedostatků aplikace. Nejdříve se zaměříme na webovou službu, u které nejdůležitějším rozšířením bude zabezpečení. Jedním ze způsobů, jak zvýšit bezpečnost aplikace, je implementace standardu WS-Security například pomocí softwarového balíčku společnosti Microsoft WSE 3.0 (Web Services Enhancements) [17] pro platformu .NET. Z pohledu dalšího vývoje webové služby, je možné doimplementovat metody pro ostatní aplikace společnosti

Google jako je Gmail, Picasa, Youtube a další. Tímto bychom vytvořili komplexní webovou službu nabízející možnost využití velkého množství aplikací pod jedním účtem Google Account. Pokud budeme hledat rozšíření klientské aplikace, určitě narazíme na možnost implementace více vláken. Jedním z řešení je tvorba jednoho vlákna pro zobrazování uživatelského rozhraní a druhého pro vykonávání operací jako je například komunikace se servery. Dalším rozšířením by bylo vytvoření propracovanějšího algoritmu na zpracování WSDL souboru. Pokud bychom chtěli, určitě najdeme mnohá další rozšíření, která by zjednodušila práci a rozšířila funkčnost této aplikace.

Díky této bakalářské práci mi byl objasněn velký potenciál webových služeb a jejich využití při strojové komunikaci přes internet. Webové služby nabízejí univerzální prostředek pro výměnu informací mezi různými koncovými body. Takováto komunikace funguje na protokolu HTTP (nebo HTTPS), který nemá problémy při průchodu sítí přes různé směrovače a tak se stává výborným prostředkem pro posílání zpráv. Další výhodou je jednoduchý formát komunikačního protokolu SOAP, který umožňuje velice jednoduchou implementaci. Právě díky těmto výhodám se stávají webové služby velice populární a jsou čím dál více využívané.

Literatura

[1] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall PTR, 2005, ISBN 0-13-185858-0.

[2] Newcomer, E.: Understanding Web Services: XML, WSDL, SOAP, and UDDI. Addison-Wesley, 2002, ISBN 0-201-75081-3.

[3] W3C. Web Services Architecture. Working group note, 2004. Dokument dostupný na http://www.w3.org/TR/ws-arch/ (duben 2008)

[4] W3C. Simple Object Access Protocol (SOAP) 1.1, W3C Note, 8. May 2000. Dokument dostupný na URL http://www.w3.org/TR/2000/NOTE-SOAP-20000508/ (duben 2008)

[5] W3C. Web Services Description Language (WSDL) Version 2.0, W3C Recommendation, 26. June 2007. Dokument dostupný na URL http://www.w3.org/TR/wsdl20/ (duben 2008)

[6] W3CSchool. WSDL Tutorial. Dostupný na http://www.w3schools.com/wsdl/default.asp (duben 2008)

[7] Zapletal, Lukáš: Protokol HTTP 1.1 pod lupou, 27. březen 2001. Dokument dostupný na URL http://www.root.cz/clanky/protokol-http-1-1-pod-lupou/ (duben 2008)

[8] Kuba, Martin: Web Services. Dokument dostupný na URL http://www.ics.muni.cz/~makub/soap/MartinKuba_WebServices_Datakon2006_clanek.pdf (duben 2008)

[9] Wikipedia. Web service. Dokument dostupný na http://en.wikipedia.org/wiki/Web_services (duben 2008)

[10] Wikipedia. SOAP. Dokument dostupný na http://en.wikipedia.org/wiki/SOAP (duben 2008)

[11] Wikipedia. Web Services Description Language (WSDL). Dokument dostupný na URL http://en.wikipedia.org/wiki/Web_Services_Description_Language (duben 2008)

[12] Wikipedia. Universal Description, Discovery and Integration (UDDI). Dokument dostupný na URL http://en.wikipedia.org/wiki/UDDI (duben 2008)

[13] Wikipedia. Hypertext Transfer Protocol (HTTP). Dokument dostupný na URL http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol (duben 2008)

[14] Wikipedia. Extensible Markup Language (XML). Dokument dostupný na URL http://en.wikipedia.org/wiki/XML (duben 2008)

[15] Google Calendar API. Dokument dostupný na URL http://code.google.com/apis/calendar/ (duben 2008)

[16] Thanh, Dao: WSDL and Schema Parser. 25. November 2005, Dokument dostupný na URL http://www.codeproject.com/KB/cpp/wsdlparser.aspx (duben 2008)

[17] MSDN. Web Services Enhacements. Dokument dostupný na URL http://msdn2.microsoft.com/en-us/webservices/aa740663.aspx (duben 2008)

Seznam příloh

Příloha 1. Webové rozhraní – metody WS

- **AddEventReminder** - This method create event reminder to selected event (param calendar ID, param event, param reminder, return reminded events). It requires to set in soap header email (userName) and password (userPassword).
- **AddRuleToAcl** - This method create Access Control List rule for selected calendar (param calendar ID, param username and ACL rule, return created rule). It requires to set in soap header email (userName) and password (userPassword).
- **CheckUserCredentials** - Method check user's name and password
- **CreateCalendar** - This method create a google calendar (Param title is mandatory. Param color and timezone must be from file timezones.xml, calendarColors.xml. Returns XmlDocument with created calendar.). It requires to set in soap header email (userName) and password (userPassword).
- **CreateEventInCalendar** - This method create an event in selected calendar (param calendar ID, param event informaitons, return created event). It requires to set in soap header email (userName) and password (userPassword).
- **CreateEventInPrimaryCalendar** - This method create an event in primary calendar (param event informaitons, return created event). It requires to set in soap header email (userName) and password (userPassword).
- **CreateRecurringEventInCalendar** - This method create a recurring event in selected calendar (param calendar ID, param recurring event informaitons, return created event). It requires to set in soap header email (userName) and password (userPassword).
- **CreateRecurringEventInPrimaryCalendar** - This method create a recurring event in primary calendar (param recurring event informaitons, return created event). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteAllEventReminders** - This method delete all event reminders in seleceted event and calendar (param calendar ID, param event, return reminded events). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteCalendar** - This method delete calendar entry (Param calendarId select deleting calendar. Returns success or error message). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteEventInCalendar** - This method delete an event in selected calendar (param calendar ID, param event info, return success message). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteEventInPrimaryCalendar** - This method delete an event in primary calendar (param calendar ID, param event info, return success message). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteRuleInAcl** - This method delete Access Control List rule for selected calendar (param calendar ID, param username, return success message). It requires to set in soap header email (userName) and password (userPassword).
- **DeleteSubscription** - This method delete subscription (param subscription identification (calendar ID)). It requires to set in soap header email (userName) and password (userPassword).
- **NewSubscription** - This method create a subsrciption to new calendar (param new calendar ID, return subscribed calendar). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveAclForCalendar** - This method download Access Control List for selected calendar (param calendar ID, return ACL). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveAllCalendars** - This method downloads all calendar entries from google calendar (returns XmlDocument with all calendars). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveEventsForCalendar** - This method download all events of certain calendar (param selected calendar, return calendar events). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveEventsForDateRange** - This method download all event of certain calendar from selected dateTime range (param calendar ID, param range start/end date, return found events). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveEventsForDateRangeAndTitleSearch** - This method download all event of certain calendar matching searching phrase (param calendar ID, param searching phrase, return found events. It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveEventsForFulltextSearch** - This method download all event of certain calendar matching searching phrase (param calendar ID, param searching phrase, return found events. It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveEventsForPrimaryCalendar** - This method download all events of certain calendar (return calendar events). It requires to set in soap header email (userName) and password (userPassword).
- **RetrieveOwnCalendars** - This method downloads own calendar entries from google calendar (returns XmlDocument with own calendars). It requires to set in soap header email (userName) and password (userPassword).
- **SearchCalendarByTitle** - This method find calendars by title (Param calendar title. Returns XmlDocument with found calendars). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateCaledar** - This method update calendar entry (Param updates calendar informations. Returns updated calendar.). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateEventInCalendar** - This method update the event in selected calendar (param old event info, param new event info, return updated event). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateEventInPrimaryCalendar** - This method update the event in primary calendar (param old event info, param new event info, return updated event). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateEventToRecurrentInPrimaryCalendar** - This method update the event to be recurring event in primary calendar (param old event info, param new event info, param recurring info, return updated event). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateRecurringEventInCalendar** - This method update the event to be recurring event in selected calendar (param calendar ID, param old event info, param new event info, param recurring info, return updated event). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateRuleInAcl** - This method update Access Control List rule for selected calendar (param calendar ID, param username and new ACL role, return updated rule). It requires to set in soap header email (userName) and password (userPassword).
- **UpdateSubscription** - This method update existing subscription (param new subscription color, param will be hidden, returns updated subscription). It requires to set in soap header email (userName) and password (userPassword).